量子回路計算量と制御 NOT ゲート数の関係について(計算機科学の理論とその応用)

AUTHOR(S):

大久保, 誠也; 青木, 輝人; 柿下, 容弓; 西野, 哲朗

CITATION:

大久保, 誠也 ...[et al]. 量子回路計算量と制御 NOT ゲート数の関係について(計算機科学の理論とその応用). 数理解析研究所講究録 2007, 1554: 125-130

ISSUE DATE:

2007-05

URL:

http://hdl.handle.net/2433/80963

数理解析研究所講究録
第1554巻 2007年 125-130

量子回路計算量と制御NOTゲート数の関係について

大久保 誠也[†], 青木 輝人[†], 柿下 容弓[†], 西野 哲朗[††]
[†]電気通信大学大学院情報通信工学専攻
[††]電気通信大学情報通信工学科

概要

本論では，量子回路計算量と制御NOTゲート（C-NOTゲート）数の関係について議論する．まずはじめに，量子回路のサイズは，含まれるC-NOTゲート数を最小化した回路のC-NOTゲート数と同じオーダーとなることを示す．次に，C-NOTゲートのみから構成される量子ビット数nの量子回路のサイズは，$O(n^2)$であることを示す．さらに，C-NOTゲートとNOTゲートのみから構成される量子ビット数nの量子回路のサイズも，$O(n^2)$であることを示す．また，その出力可能なパターン数はC-NOTゲートのみで構成された回路が出力できるパターン数の高々2^n倍であることを示す．

1 はじめに

1985年にD. Deutschが，量子力学に基づく新たな計算モデルとして量子Turing機械を提案し，量子計算機のモデル化を行って以来，量子計算に関する研究が活発に行われてきた．例えば，1994年にP. W. Shorは，整数の因数分解を多項式時間内に高い成功確率で行う量子アルゴリズムを示した[3]．さらに，1996年にはL.K.Groverが，効率的量子探索アルゴリズムを提案した[1]．このように，量子計算は本質的に古典計算よりも強力である可能性がある．

また，一方，幾何学的手法を用いた量子論理回路のサイズの下界に関する研究や，補助量子ビットが回路計算量に及ぼす影響を明らかにする研究が行われている[2, 5]．これらの研究により，通常の計算量理論に，何らかの貢献ができるのではないかと期待されている．

量子計算機の物理的実装の実現には，多くの困難が存在する．例えば，量子もつれ合いを保つことができる時間が短いことや，複雑な量子操作を行なうことは難しい等である．これらの問題の解決のためにも，量子回路サイズは重要である．

本研究では，量子回路サイズを評価するよい方法を検討する．特に，C-NOTゲートに着目し，量子回路計算量とC-NOTゲート数の関係について，幾つかの定理を示す．

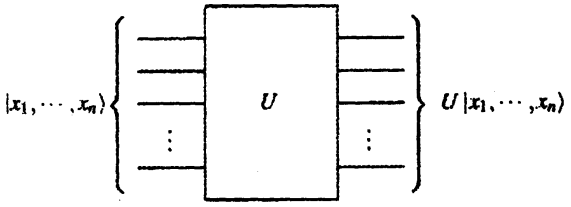

図1: 量子回路

2 諸定義

量子計算は，ベクトル空間$\underbrace{\mathbb{C}^2\otimes\cdots\otimes\mathbb{C}^2}_{n}$上のユニタリ変換$U$を実行する**量子回路**として定義される．$n$量子ビットの量子回路は，入力$|x_1,\cdots,x_n\rangle$に対し，$2^n\times 2^n$ユニタリ変換$U$を適用し，$U|x_1,\cdots,x_n\rangle$を出力する．また，このような量子回路を図1のようなダイアグラムで表す．ダイアグラムにおいて，平行したn本のワイヤーはベクトル空間$\underbrace{\mathbb{C}^2\otimes\cdots\otimes\mathbb{C}^2}_{n}$を表し，1本のワイヤーが1量子ビットに相当する．

任意の量子回路は基本量子ゲートを組み合わせることで構成することができる[4]．また，量子回路のサイズとは，基本量子ゲートの数のことである．本研究では，**C-NOTゲートと1量子**

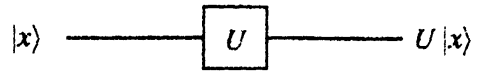

図 2: 1 量子ビットゲート

図 3: NOT ゲート

ビットゲートを基本量子ゲートとして用いる．ここで，C-NOT ゲートと 1 量子ビットゲートとは，次のようなゲートである．

定義 1 (1 量子ビット ゲート) U をベクトル空間 $\mathbb{C}^2$ 上の任意のユニタリ作用素であるとする．入力 $|x\rangle$ に対して，$U|x\rangle$ を出力する量子ゲートを，1 量子ビットゲートという．また，図 2 のようなダイアグラムで表記する．

特に 1 量子ビットの NOT を計算する 1 量子ビットゲートを **NOT ゲート**とよび，図 3 のように表記する．

定義 2 (C-NOT ゲート) 入力 $|x_1\rangle, |x_2\rangle \in \{|0\rangle, |1\rangle\}$ に対して，$|x_1\rangle|x_1 \oplus x_2\rangle$ を出力するゲートを**制御 NOT ゲート（C-NOT ゲート）**という．また，x_1 を出力している方のビットを制御ビット，$x_1 \oplus x_2$ を出力している方のビットを目標ビットという．図 4 のようなダイアグラムで表記する．

3 量子回路のサイズと C-NOT ゲート数の関係

本節では，量子回路のサイズと C-NOT ゲート数の関係について考察する．

あるユニタリ作用素 U を実現するにあたり，回路サイズが最小になる回路のゲート数を $m_G(U)$ と，また，C-NOT ゲート数が最小となる回路の C-NOT ゲート数を $m_C(U)$ と表記する．

定理 1 $m_C(U) = \Omega(n)$ のとき，次の関係が成立

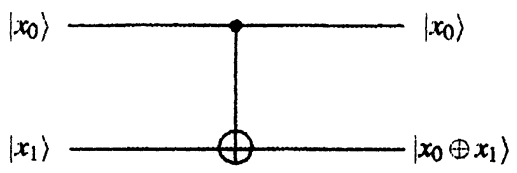

図 4: C-NOT ゲート

する．

$$m_G(U) = \Theta(m_C(U))$$

証明

1. $m_G(U) = \Omega(m_C(U))$ の証明
   回路サイズが最小となる回路の C-NOT ゲート数を c'，1 qubit ゲートの数を s' とすると，

   $$m_G(U) = c' + s'$$

   が成立する．あきらかに $m_C(U) \leq c', s' \geq 0$ なので，

   $$m_C(U) \leq m_G(U)$$

   が成り立つ．したがって，

   $$m_G(U) = \Omega(m_C(U)) \qquad (1)$$

   となる．
2. $m_G(U) = O(m_C(U))$ の証明
   C-NOT ゲート数が最小になる回路の回路サイズ g を

   $$g = m_C(U) + s$$

   とする．ここで，s は 1 量子ビットゲートの個数である．同じワイヤー上で隣接する 1 量子ビットゲートは，ひとつの 1 量子ビットゲートにまとめることができる．したがって，量子回路に含まれる 1 量子ビットゲートは，C-NOT ゲートのすぐ右隣に 2 つと，n 本ある各ワイヤーの最も左に 1 つずつあれば十分である．よって，

   $$g \leq 3m_C(U) + n$$

   が成立する．あきらかに，$m_G(U) \leq g$ なので

   $$m_G(U) \leq 3m_C(U) + n$$

が成り立つ．よって，$m_C(U)=\Omega(n)$ のとき，

$$m_G(U)=O(m_C(U)) \qquad (2)$$

となる．

式 (1),(2) より $m_C(U)=\Theta(m_G(U))$ を得る．□

4 ゲートの種類を制限した場合

本節では，使用するゲートの種類を制限した場合における回路サイズについて，考察を行なう．

定理 2 C-NOT ゲートのみから構成される量子ビット数 n の量子回路のサイズは，高々 $O(n^2)$ である．

証明 C-NOT ゲートのみを利用した回路においては，第 1 番目から第 n 番目までの，各々のワイヤーから出力される値は，入力 x_i の排他的論理和となる．各ワイヤーからの出力が x_i の排他的論理和の形で与えられたとき，C-NOT ゲートのみを使用した回路を構成するアルゴリズムを示すことで，定理の証明を行なう．

第 i 番目のワイヤーからの出力を $|z_{i,1}x_1 \oplus z_{i,2}x_2 \oplus \cdots \oplus z_{i,n}x_n\rangle$, $z_{i,j} \in \{0,1\}$ としたとき，すべてのワイヤーからの出力をまとめて，

$$Z = \begin{pmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,n} \\ z_{2,1} & z_{2,2} & & z_{2,n} \\ \vdots & & & \vdots \\ z_{n,1} & z_{n,2} & \cdots & z_{n,n} \end{pmatrix}$$

のように表現する．行列 Z のランクが n で無い場合，その回路は C-NOT ゲートのみで構成することはできない．

回路構成アルゴリズム

入力：Z

出力：*C−NOT* ゲートのみを使用した回路

アルゴリズム：

1. ゲートが 1 つもない（つまりワイヤーのみの）量子回路を書く．
2. 2a から 2b を $i=1$ から n まで繰り返す．

(a) もし，$z_{i,i}=0$ ならば，$z_{j,i}=1$ である列 j を 1 つ探し出し，$z_{i,k} := z_{i,k} \oplus z_{j,k}, k \in \{1,\cdots,n\}$ とする．
すなわち，第 i 行目と第 j 行目の，それぞれの要素の排他的論理和を，新たな第 i 行目の要素とする．
第 j 番目のワイヤーを制御ビット，第 i 番目のワイヤーを目標ビットとした C-NOT ゲートを，量子回路の一番左側に置く．

(b) $j=1$ から n（ただし $j=i$ は除く）に対して，以下を繰り返す．

i. $z_{j,i}=1$ ならば，
$z_{j,k} := z_{i,k} \oplus z_{j,k}, k \in \{1,\cdots,n\}$
とする．
すなわち，第 i 行目と第 j 行目の，それぞれの要素の排他的論理和を，新たな第 j 行目の要素とする．
第 i 番目のワイヤーを制御ビット，第 j 番目のワイヤーを目標ビットとした C-NOT ゲートを，量子回路の一番左側に置く．

ii. $z_{j,i}=0$ ならば，何もしない．

アルゴリズムの実行例を Appendix.A に示す．

ステップ 2a において，C-NOT ゲートは高々 1 個，ステップ 2b において，C-NOT ゲートは高々 $n-1$ 個記入される．また，ステップ 2a から 2b は，n 回繰り返されるので，全体として，書き込まれる C-NOT ゲートの個数は，高々 n^2 個である．□

定理 3 C-NOT ゲートと NOT ゲートのみから構成される量子ビット数 n の量子回路のサイズは，高々 $O(n^2)$ である．

また，C-NOT ゲートと NOT ゲートのみで構成された回路が出力できるパターン数は，C-NOT ゲートのみで構成された回路が出力できるパターン数の，高々 2^n 倍である．

証明

C-NOT ゲートと NOT ゲートのみから構成される回路においては，それぞれのワイヤーからの出力は，リテラルの排他的論理和と否定のみで構成された式となる．

また，排他的論理和と否定のみで構成された式においては，

$$\overline{x \oplus y} = \bar{x} \oplus y = x \oplus \bar{y}$$

が成り立つ．つまり，1つのリテラルの否定は全体の否定と等価である．リテラルが3つ以上含まれる場合でも，1つのリテラルの否定は全体の否定と等価である．したがって，リテラルの排他的論理和と否定のみで構成可能な式は，排他的論理和のみで構成される形の式と，排他的論理和のみの式全体を否定した形の式の，2パターンしかない．

これらのことより，出力に対応する排他的論理和の式中に否定が含まれていた場合，その回路の最後の層にNOTゲートを置くことで，その層の直前においては，排他的論理和のみの式とすることができる（図5参照）．

排他的論理和のみの式を計算するのに必要な回路サイズは，$O(n^2)$であり，また，この回路に含まれるNOTゲートの数は，高々n個である．したがって，C-NOTゲートとNOTゲートのみを使用して量子回路を組む場合，必要なゲート数は高々$O(n^2)$である．

C-NOTゲートとNOTゲートのみによって構成される回路は，C-NOTゲートのみによって構成される回路と比べ，それぞれのワイヤーの最後の層に，NOTゲートが有るか否かの差しかない．そのため，C-NOTゲートとNOTゲートしか使用しない回路が出力できるパターン数は，C-NOTのみの回路が出力できるパターン数の高々2^n倍となる．□

5 おわりに

本論では，量子回路計算量と制御NOTゲート数の関係について議論を行なった．

これらの議論より，量子回路計算量の評価においては，C-NOTゲート数が本質的であることがわかったが，一方，C-NOTゲートやNOTゲートのみでは，単純な回路しか構成できないことも判明した．

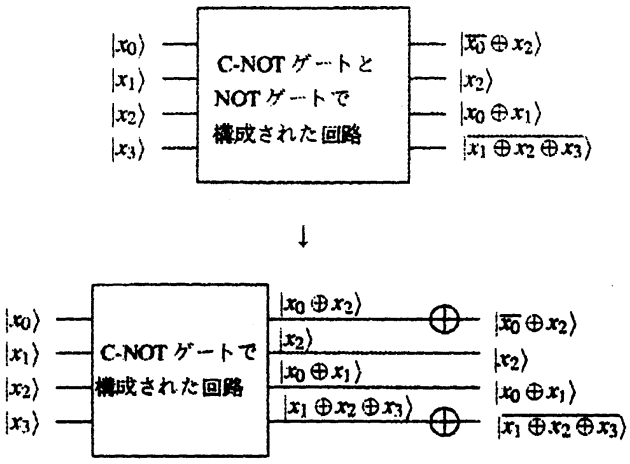

図5: C-NOTゲートとNOTゲートのみによって構成される回路

今後の課題としては，C-NOTゲートとある特定の1量子ビットゲートのみからなる回路のサイズを評価することがあげられる．

参考文献

[1] Grover, L.: Quantum Mechanics Helps in Searching for a Needle in a Haystack, *Physical Review Letters*, Vol. 79, No. 2, pp. 325–328 (1997).

[2] Nielsen, M.: A geometric approach to quantum circuit lower bounds, quant-ph/*0502070* (2005).

[3] Shor, P.: Algorithms for Quantum Computation : Discrete Log and Factoring, *in Proceedings of the 35th Anniual IEEE Symposium on Foundations of Computer Science* (1994).

[4] 上坂吉則: 量子コンピュータの基礎数理, コロナ社 (2000).

[5] 青木輝人, 大久保誠也, 西野哲朗: 量子回路における補助量子ビットの効果について, 2006年度夏のLAシンポジウム (2006).

A アルゴリズムの実行例：

$|x_1,\cdots,x_4\rangle$ を入力したとき，$|x_1 \oplus x_3\rangle\,|x_3\rangle\,|x_1 \oplus x_2\rangle\,|x_2 \oplus x_3 \oplus x_4\rangle$ を出力する量子回路を構成する．

1. （アルゴリズム中，ステップ 1）4 入力・4 出力の量子回路なので，4 本のワイヤーを書く．また，行列表現は

$$Z = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

となる（図 6 参照）．

2. （アルゴリズム中，ステップ 2）$i = 1$ とする．
3. （アルゴリズム中，ステップ 2a）$z_{1,1} = 1$ なので，ステップ 2b に進む．
4. （アルゴリズム中，ステップ 2b）$z_{3,1} = 1$ なので，第 1 行目と第 3 行目の各々の要素の排他的論理和を，新たな第 3 行目とする．第 1 番目のワイヤーを制御ビット，第 3 番目のワイヤーを目標ビットとした C-NOT ゲートを置く（図 7 参照）．
   他に $z_{j,i} = 1, j \neq i$ は無いので，ステップ 2 の初めに戻る．
5. （アルゴリズム中，ステップ 2）$i = 2$ とする．
6. （アルゴリズム中，ステップ 2a）$z_{2,2} = 0$ である．$z_{3,2} = 1$ であるので，第 2 行目と第 3 行目の各々の要素の排他的論理和を，新たな第 2 行目とする．第 3 番目のワイヤーを制御ビット，第 2 番目のワイヤーを目標ビットとした C-NOT ゲートを置く（図 8 参照）．

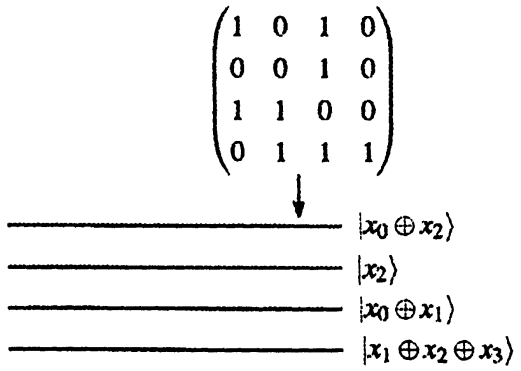

図 6: 初期状態

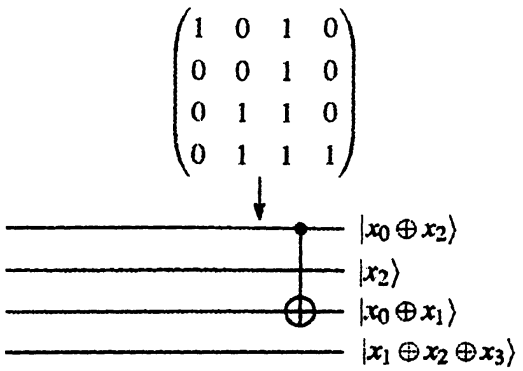

図 7: 例中のステップ 4 終了後

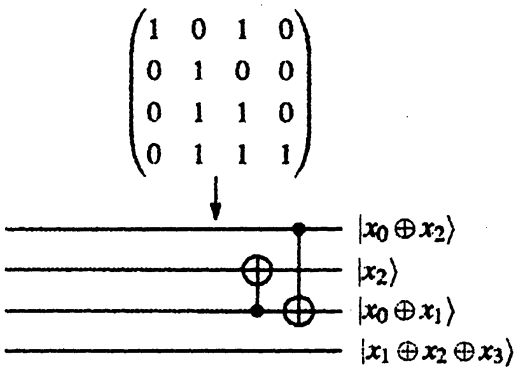

図 8: 例中のステップ 6 終了後

7. （アルゴリズム中，ステップ 2b）$z_{3,2} = 1$ なので，第 2 行目と第 3 行目の各々の要素の排他的論理和を，新たな第 3 行目とする．第 2 番目のワイヤーを制御ビット，第 3 番目のワイヤーを目標ビットとした C-NOT ゲートを置く（図 9 参照）．
8. （アルゴリズム中，ステップ 2b）$z_{4,2} = 1$ なので，第 2 行目と第 4 行目の各々の要素の排他的論理和を，新たな第 4 行目とする．第 2 番目のワイヤーを制御ビット，第 4 番目のワイヤーを目標ビットとした C-NOT ゲートを置く（図 10 参照）．
   他に $z_{j,i} = 1, j \neq i$ は無いので，ステップ 2 の初めに戻る．
9. （アルゴリズム中，ステップ 2）$i = 3$ とする．
10. （アルゴリズム中，ステップ 2a）$z_{3,3} = 1$ なので，ステップ 2b に進む．

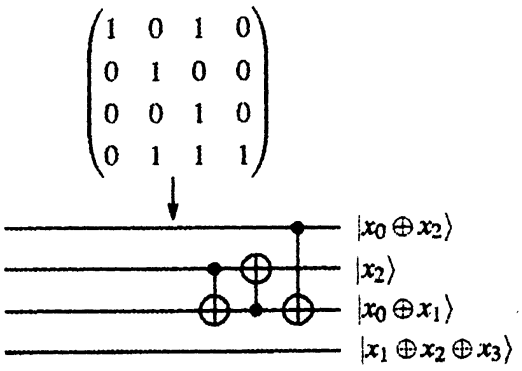

図 9: 例中のステップ 7 終了後

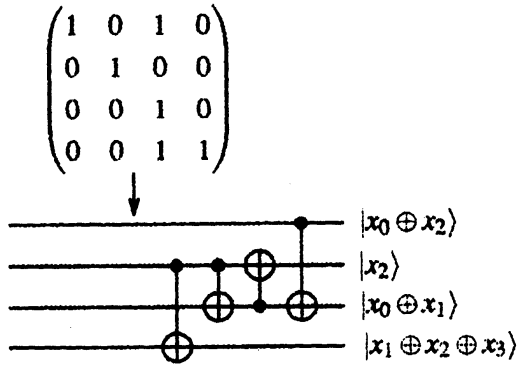

図 10: 例中のステップ 8 終了後

11. （アルゴリズム中，ステップ 2b）$z_{1,3} = 1$ なので，第 1 行目と第 3 行目の各々の要素の排他的論理和を，新たな第 1 行目とする．第 3 番目のワイヤーを制御ビット，第 1 番目のワイヤーを目標ビットとした C-NOT ゲートを置く（図 11 参照）．
12. （アルゴリズム中，ステップ 2b）$z_{4,3} = 1$ なので，第 3 行目と第 4 行目の各々の要素の排他的論理和を，新たな第 4 行目とする．第 3 番目のワイヤーを制御ビット，第 4 番目のワイヤーを目標ビットとした C-NOT ゲートを置く（図 12 参照）．
    他に $z_{j,i} = 1, j \neq i$ は無いので，ステップ 2 の初めに戻る．
13. （アルゴリズム中，ステップ 2）$i = 4$ とする．
14. （アルゴリズム中，ステップ 2a）$z_{4,4} = 1$ なので，ステップ 2b に進む．
15. $z_{j,i} = 1, j \neq i$ は無い．
16. $i = 1$ から 4 まで終了したので，現在の回路を出力して停止する（図 13 参照）．

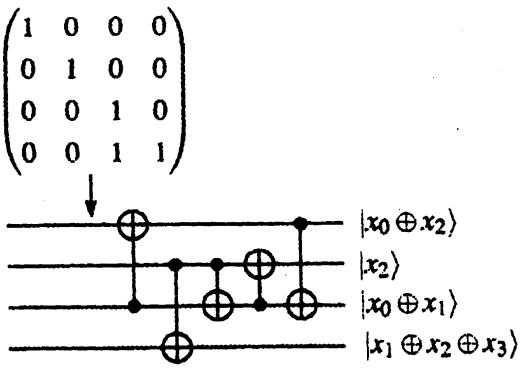

図 11: 例中のステップ 11 終了後

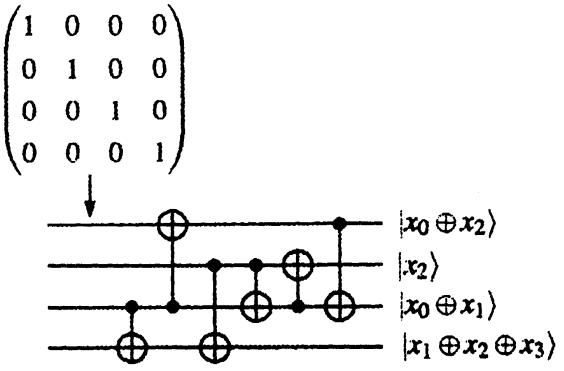

図 12: 例中のステップ 12 終了後

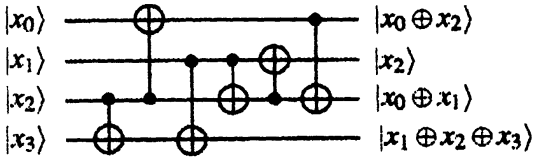

図 13: 最終的に得られた回路